AI Knowledge Assignment

AI Knowledge Discovery Exercise: RAG Architecture & Unstructured Data Analysis

Assignment Overview

Duration: 4-6 weeks
Objective: Explore how AI can leverage unstructured organizational knowledge through Retrieval-Augmented Generation (RAG) architectures and develop practical understanding of knowledge extraction from various enterprise systems.

Learning Objectives

By the end of this exercise, you will:

- Understand RAG architecture fundamentals and variants
- Build and test different RAG implementations
- Evaluate knowledge extraction from structured and unstructured sources
- Create a functional prototype with real-time knowledge access
- Provide recommendations for enterprise-scale implementation

Part 1: RAG Architecture Foundation (Week 1)

1.1 Theoretical Understanding

Goal: Build foundational knowledge of RAG systems

Key Concepts to Research:

- **Retrieval-Augmented Generation (RAG):** How it combines retrieval and generation
- **Vector Embeddings:** How text is converted to numerical representations
- **Semantic Search:** Finding relevant content based on meaning, not just keywords
- **Chunk Strategies:** How to break documents into searchable pieces

- **Vector Databases:** Storage and retrieval of embeddings

Recommended Resources:

1. Read: "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" (original RAG paper)
2. Watch: Langchain RAG tutorials on YouTube
3. Explore: OpenAI embeddings documentation
4. Review: Pinecone/Weaviate/Chroma vector database documentation

Deliverable:

Write a 2-page summary explaining RAG in simple terms, including diagrams showing the flow from user query to response.

1.2 RAG Architecture Variants

Research and document these different approaches:

Basic RAG

- Simple retrieval → context injection → generation
- Best for: Straightforward Q&A scenarios

Advanced RAG

- **Multi-step retrieval:** Query expansion, re-ranking
- **Fusion retrieval:** Combining multiple search strategies
- **Contextual compression:** Filtering retrieved content for relevance

Agentic RAG

- **Tool-calling:** RAG system can decide when to search vs. generate
- **Multi-source:** Accessing different knowledge bases based on query type
- **Iterative refinement:** System can ask follow-up questions

Deliverable:

Create a comparison matrix of RAG variants with pros/cons and use cases.

Part 2: Technical Implementation (Weeks 2-3)

2.1 Environment Setup

Set up your development environment:

```
# Required tools and libraries
- Python 3.9+
- Jupyter Notebook or VS Code
- Required packages:
  * langchain
  * openai
  * chromadb (or pinecone/weaviate)
  * streamlit (for UI)
  * pandas
  * pypdf2
  * python-docx
  * requests
```

2.2 Build Basic RAG System

Create a simple RAG implementation:

Step 1: Document Processing Pipeline

```
# Your pipeline should handle:
1. PDF extraction (PyPDF2, pdfplumber)
2. Word document processing (python-docx)
3. Excel file processing (pandas)
4. Text chunking strategies:
   - Fixed size (500-1000 tokens)
   - Semantic chunking (paragraph/section breaks)
   - Sliding window with overlap
```

Step 2: Embedding and Storage

```
# Implement multiple embedding strategies:
1. OpenAI ada-002 embeddings
2. Sentence-transformers (local)
3. Compare embedding quality for different content types
```

Step 3: Retrieval System

```
# Build retrieval with:
1. Similarity search (cosine similarity)
2. Hybrid search (keyword + semantic)
3. Metadata filtering (document type, date, department)
```

Step 4: Generation Component

```
# Integrate with LLM:
1. OpenAI GPT models
2. Prompt engineering for context utilization
3. Citation and source tracking
```

2.3 User Interface Development

Build a Streamlit app with:

File Upload Interface

- Support for PDF, DOCX, XLSX, TXT files
- Batch upload capability
- Progress indicators for processing

RAG Configuration Panel

- Chunk size adjustment
- Embedding model selection
- Retrieval method selection
- Number of retrieved documents

Chat Interface

- Real-time query processing

- Source document citations
- Response quality feedback

Admin Dashboard

- Document inventory
- Search analytics
- Performance metrics

Deliverable:

Working Streamlit application with code documentation.

Part 3: Knowledge Source Integration (Week 3-4)

3.1 Structured Data Sources

Explore integration with enterprise systems:

Salesforce Knowledge Base

```
# Research and prototype:
1. Salesforce API authentication
2. Knowledge article extraction
3. Metadata preservation (categories, tags, update dates)
4. Incremental sync strategies
```

SharePoint Integration

```
# Investigate:
1. SharePoint REST API / Graph API
2. Document libraries access
3. Permission handling
4. Version control considerations
```

Database Integration

```
# Consider:
1. FAQ databases
2. Product catalogs
3. Policy documents
4. Training materials
```

3.2 Unstructured Data Challenges

Address real-world data issues:

File System Crawling

- Directory traversal strategies
- File type detection and filtering
- Duplicate document handling
- Large file processing

Content Quality Issues

- Scanned documents (OCR requirements)
- Mixed language content
- Legacy file formats
- Corrupted or password-protected files

Metadata Extraction

- File properties (creation date, author, department)
- Content analysis for automatic tagging
- Business context identification

3.3 Data Pipeline Architecture

Design scalable ingestion:

Batch Processing

- Scheduled document processing
- Delta updates (only new/changed files)
- Error handling and retry logic

Real-time Processing

- File system monitoring
- API webhooks for system updates
- Streaming data integration

Deliverable:

Technical architecture document with data flow diagrams and integration prototypes.

Part 4: Testing & Evaluation (Week 4-5)

4.1 Create Test Dataset

Prepare diverse content for testing:

Document Types

- Technical documentation
- Policy documents
- Meeting minutes
- Email threads
- Presentation slides
- Spreadsheet data

Test Queries

- Factual questions (direct answers in documents)
- Analytical questions (requiring synthesis)
- Comparison questions (multiple document analysis)
- Procedural questions (step-by-step processes)

4.2 Evaluation Metrics

Measure system performance:

Retrieval Quality

- Precision@K (relevant docs in top K results)

- Recall (relevant docs found vs. total relevant)
- MRR (Mean Reciprocal Rank)

Generation Quality

- Answer accuracy (manual evaluation)
- Citation accuracy (sources correctly referenced)
- Response completeness
- Hallucination detection

System Performance

- Query response time
- Document processing speed
- Storage efficiency
- Concurrent user handling

User Experience

- Query success rate
- User satisfaction scores
- Task completion time

4.3 A/B Testing Framework

Compare different approaches:

Chunking Strategies

- Fixed vs. semantic chunking
- Chunk size optimization
- Overlap strategies

Retrieval Methods

- Pure semantic search
- Hybrid keyword + semantic
- Re-ranking approaches

Generation Approaches

- Different prompt templates
- Context length optimization
- Temperature and creativity settings

Deliverable:

Comprehensive evaluation report with performance benchmarks and recommendations.

Part 5: Enterprise Considerations (Week 5-6)

5.1 Scalability Analysis

Assess enterprise readiness:

Technical Scalability

- Vector database performance at scale
- Concurrent query handling
- Document processing throughput
- Storage requirements and costs

Operational Scalability

- Content governance workflows
- User access control
- System monitoring and maintenance
- Update and refresh procedures

5.2 Security & Compliance

Address enterprise concerns:

Data Security

- Document access permissions
- Query logging and auditing
- Data encryption (at rest and in transit)
- User authentication and authorization

Compliance Requirements

- Data retention policies
- Privacy considerations (PII handling)
- Audit trail requirements
- Regulatory compliance (if applicable)

5.3 Integration Strategy

Plan for organizational adoption:

Change Management

- User training requirements
- Adoption strategy
- Success metrics and KPIs
- Feedback collection mechanisms

Technical Integration

- API design for existing systems
- Single sign-on integration
- Workflow automation possibilities
- Mobile access considerations

Deliverable:

Enterprise implementation roadmap with risk assessment and mitigation strategies.

Part 6: Final Deliverables & Recommendations

6.1 Prototype Demo

Prepare comprehensive demonstration:

- Working RAG system with multiple knowledge sources
- Live query demonstration
- Performance metrics presentation
- Architecture overview

6.2 Technical Documentation

Create implementation guides:

- System architecture documentation
- API documentation
- Deployment instructions
- Troubleshooting guide

6.3 Business Case

Develop recommendation report:

- ROI analysis and cost considerations
- Implementation timeline
- Resource requirements
- Risk assessment
- Success metrics and measurement plan

6.4 Future Research Directions

Identify next steps:

- Advanced RAG techniques to explore
- Integration opportunities
- Potential pilot programs
- Technology evolution considerations

Resources & Tools

Development Tools

- **Code Editor:** VS Code with Python extension
- **Notebooks:** Jupyter Lab for experimentation
- **Version Control:** Git for code management
- **Documentation:** Markdown for technical docs

AI/ML Libraries

- **LangChain:** RAG framework and components
- **LlamaIndex:** Alternative RAG framework
- **Transformers:** Hugging Face model library
- **OpenAI:** API for embeddings and chat models

Vector Databases

- **ChromaDB:** Local development and testing
- **Pinecone:** Cloud-based vector database
- **Weaviate:** Open-source vector database
- **FAISS:** Facebook's similarity search library

UI Frameworks

- **Streamlit:** Rapid prototyping interface
- **Gradio:** Alternative UI framework
- **React:** For production interfaces

Enterprise Integration

- **Salesforce APIs:** REST and GraphQL
- **Microsoft Graph:** SharePoint and Office 365
- **AWS/Azure:** Cloud deployment options

Success Criteria

Technical Success

- [ ] Functional RAG system with multiple document types
- [ ] UI allowing file upload and real-time querying
- [ ] Integration with at least one structured data source
- [ ] Performance benchmarks and evaluation metrics
- [ ] Scalable architecture design

Learning Success

- [ ] Clear understanding of RAG principles and variants

- [ ] Practical experience with vector embeddings and databases
- [ ] Knowledge of enterprise integration challenges
- [ ] Ability to evaluate and compare different approaches
- [ ] Business-ready recommendations and implementation plan

Deliverable Success

- [ ] Working prototype with documentation
- [ ] Comprehensive evaluation report
- [ ] Enterprise implementation roadmap
- [ ] Technical architecture documentation
- [ ] Business case with ROI analysis

Weekly Check-ins

Schedule weekly progress reviews to discuss:

- Technical challenges and solutions
- Learning insights and questions
- Prototype development progress
- Enterprise considerations and requirements
- Timeline adjustments if needed

Additional Support

Don't hesitate to reach out for:

- Technical guidance on implementation details
- Business context for enterprise requirements
- Access to additional resources or tools
- Clarification on objectives or deliverables

This exercise is designed to be exploratory and educational - embrace the learning process and document both successes and challenges along the way.